OIPE JC139
NOV 24 2000
PATENT & TRADEMARK OFFICE

[Name of Document] Specification

[Title of the Invention]

METHOD OF RETRIEVAL BY A PARALLEL DATABASE SYSTEM

RECEIVED
NOV 29 2000
Group 2100

[What is claimed is]

[Claim 1]

A method of retrieval by a parallel database system comprising a front end server for analyzing a query issued to a database, and a plurality of database operation servers for operating databases, said front end server and said plurality of database operation servers being connected to each other through a network:

Wherein, for a configuration in which said database operation servers retrieve data and said front end server processes or controls a retrieval result, said database operation server returns data positional information as the retrieval result to said front end server, when retrieving data comprising a plurality of pieces of subdata; and when using each subdata of the data comprising the plurality of pieces of subdata for result processing, controlling, or database operations, which is executed after said retrieval, said front end server fetches each subdata of the data stored in said database operation server, depending on said data positional information, dictionary information on the subdata position in the data, and an identifier of each subdata required for a query, and executes a processing

using the fetched subdata.

[Claim 2]

The method of retrieval by a parallel database system according to Claim 1, wherein said data positional information includes an identifier of said database operation server, and an address of the data stored in said database operation server.

[Claim 3]

The method of retrieval by a parallel database system according to Claim 1, wherein said dictionary information on positions of respective subdata in the data includes offset values from a starting address by which the positions of respective subdata of said data clustered are represented.

[Claim 4]

The method of retrieval by a parallel database system according to Claim 1, wherein said dictionary information on positions of respective subdata in the data includes an identifier of the subdata and offset values from a starting address by which the positions of respective subdata of said data clustered are represented, and when fetching said subdata, said front end server fetches an offset value from the identifier of said subdata and fetches each subdata using the fetched offset value.

[Claim 5]

The method of retrieval by a parallel database system according to Claim 1, wherein, when the processing using the fetched subdata is an update processing for retried data, said front end server transfers an internal format processing procedure for data updating, together with said data positional information and said dictionary information on positions of said respective subdata, to said database operation server, and said internal format processing procedure for data updating uses said subdata on said database operation server side.

[Claim 6]

A method of retrieval by a parallel database system comprising a front end server for analyzing a query issued to a database, and a plurality of database operation servers for operating databases, said front end server and said plurality of database operation servers being connected to each other through a network:

Wherein, for a configuration in which said database operation servers retrieve data and said front end server processes or controls a retrieval result, said front end server selects one of the following methods in a process of analysis, depending on a predefined selection criteria, and creates/executes an internal format processing procedure corresponding to a selected method:

(1) a first method in which said database operation

server returns data positional information as the retrieval result to said front end server, when retrieving data comprising a plurality of pieces of subdata; and when using each subdata of data comprising the plurality of pieces of subdata for result processing, controlling, or database operations, which is executed after said retrieval, said front end server fetches each subdata of the data stored in said database operation server, depending on said data positional information, a dictionary information on the subdata position in the data, and an identifier of each subdata required for query, and executes a processing using the fetched subdata; and

(2) a second method in which said database operation server returns data including subdata as the retrieval result to said front end server, when retrieving data comprising a plurality of pieces of subdata, and fetches and processes said subdata on the front end server side using result processing, controlling, or database operations, which is executed after said retrieval.

[Claim 7]

The method of retrieval by a parallel database system according to Claim 5, wherein said predefined selection criteria is that costs of the methods are calculated from said dictionary information including a subdata length of each subdata and a proper method can be

selected by comparing the calculated costs with each other.

[Claim 8]

The method of retrieval by a parallel database system according to Claim 5, wherein said predefined selection criteria is that said first method should be selected when said subdata includes data longer than a standard value set by the system and said second method should be selected when said subdata does not include data longer than a standard value set by the system.

[Claim 9]

The method of retrieval by a parallel database system according to Claim 5, wherein said front end server returns a retrieval result to a user application side after said database operation server retrieves data comprising a plurality of pieces of subdata;

wherein said retrieval result includes flag information indicating that data has been transferred to said front end server or that a positional information has been transferred to said front end server without said data being transferred;

wherein said front end server creates said internal format processing procedure corresponding to said first method or said second method from said flag information, when using said subdata for result processing, controlling, or database operations, which is executed after said

retrieval.

[Detailed Description of the Invention]

[Technical Field to which the Invention Pertains]

This invention relates to a database management system, and more particularly to a query parallel processing preferable to a parallel query processing suitable for a relational database system.

[Prior Art]

As for the prior arts related to this invention, an SQL3 Abstract Data Type and a parallel database system are described as follows.

First, the following provides an explanation for the SQL3 Abstract Data Type.

Recently, a relational database, above all, an SQL database system has been applied primarily to a business data processing. In addition, an object database has been actually utilized, an object of which is to manipulate data with a complicated organization, which is difficult to efficiently manipulate in a frame of the conventional relational database.

On the other hand, a research has been promoted that the data with a complicated organization may be manipulated by extending a relational database, and SQL3 standardization has been promoted. SQL3 can manipulate a

user defined data (type) with a complicated organization, i.e., Abstract Data Type (abstract data type, hereafter abbreviated as "ADT"). ADT can manipulate complicated data adopting an object-oriented conception that a plurality of pieces of data called "attribute" (hereafter called "subdata") can be hidden with a function interface and an inheritance can be maintained between the types. A type must be defined by using a CREATE TYPE definition statement as an ADT SQL definition statement. The defined type can be used in declaring a variable or defining a row in a table, similarly to the system defined types such as an integer type and character type. Depending on each of these types, data with a complicated organization can be created and used. For details of SQL3 or ADT, refer to "Object Query Standards," "ACM SIGMOD Record, Vol. 25, No. 1, pp 87-92, March 1996" written by Andrew E. Wade, Ph.D. In addition, a draft of SQL3 standardization is contained in "ISO/IEC JTC1/SC21/WG3 DBL-MCI-004, ISO Working Draft Database Language SQL, 1996."

Secondly, the following provides an explanation for a parallel database system.

In a relational database system, it is easy to divide and allocate data into/to a plurality of database operation servers and parallel-access the database operation servers to enhance the performance. A demand for

this parallel database system has been expanding with an increase of data volume. For details of the parallel database system, refer to "Parallel Database Systems: The Future of High Performance Database Systems," CACM, Vol. 35, No. 6, 1992, written by DeWitt, D., et. al.

A parallel database system comprises a server (called "front end server") that analyzes and compiles a query issued from a user application program (called "UAP") in a host computer to a database, and a plurality of servers (called "database operation servers") that access a disk unit where data is stored and manipulate the data. The following provides a simplified description based on a system configuration comprising a host, a front end server, and a plurality of database operation servers. However, a plurality of front end servers can manipulate a plurality of queries issued form one or more host computers. Even in this case, each query may be processed in a system configuration comprising a host, a front end server, and a plurality of database operation servers. Thus, this case may be applied to the following description or this invention without a problem.

Generally, SQL as a query issued to a database (hereafter called "database query") is frequently used as embedded in a computer language (hereafter called "parent language") such as the C-language (that is hereafter called

"embedded SQL"). The parent language in a host computer issues a database query of retrieval to a database system, and data update, deletion, or insertion from/to a database. Then, the database system analyzes and executes the query, and returns the execution result to the host computer. The parent language in the host receives the execution result and uses for the result processing or control processing including condition determination, assignment, or calculation. Like a stored procedure, this invention can be applied to even the case where a database query including the control processing and result processing is issued. (In this case, the processes executed by the database operation servers including data retrieval, insertion, update, and deletion may be called "database operation statements," discriminating from the processes executed by the front end server including the control processing and result processing.) For details of the stored procedure, refer to "How to Master the Stored Procedure and Trigger, NIKKEI OPEN SYSTEMS, No. 2, pp. 133-144, 1993" written by Hatsuko Katayama.

A plurality of database queries can be embedded in a parent language, and a result can be transmitted through a variable of the parent language. Transmission of a value through the variable depends on how the parent language processes the analyzed result. When an area for storing a

value is determined for each variable and the value of the variable is used, a method of setting the variable binding to view the area may be considered.

A general example of creating, transferring, and executing a procedure of processing an internal format of a parallel database system with an embedded SQL is explained as follows. UAP control statements written by the embedded SQL execute processing or controlling the result obtained by operating a database. For the database query, a database query is sent to the front end server through a network one statement after another. Then, a compiler executes syntax analysis, semantic analysis, optimization, or compilation to create an internal format processing procedure for actual database operation based on the issued database query. The internal format processing procedure is represented by a code interpreted/executed by an interpreter or execution form code. The definition information required for compilation is stored as a dictionary information that can be accessed by the front end server. The front end server transfers the created processing procedure through the network to a database operation server that actually performs database operation, and the database operation server executes the procedure by using a start request.

The database operation server that actually performs

database operation is determined by information on segmenting a table to be usually operated. The information about segmenting the table is specified by the table definition and contained in a dictionary. The database operation server comprises a processor and one or more disk units. An improved idea has been proposed that, by storing the internal format processing procedure in a cache in the database operation server, the internal format processing procedure in the cache should be used for the second or subsequent executions according to an execution request issued.

A parallel database system comprises a plurality of database operation servers, which enable the SQL processing to be executed in parallel. The SQL processing result is transmitted between other database operation servers through the network, if necessary. Finally, after receiving the result from the UAP, the front end server returns the result to the UAP, and the UAP processes/controls the result. Thereafter, the same processings are repeated for each SQL statement.

[Problems to be Solved by the Invention]

If the data that can be manipulated by a database query is a set of a plurality of pieces of data (i.e., attribute processed by the ADT, hereafter called "subdata"), two data processing methods may be considered when

retrieving, updating, inserting, processing, or controlling the data. That is, one is that individual subdata is separately processed, and the other is that the entire data, i.e., a set of subdata is processed. As a method of using a query issued to a database, the following method is considered. First, the entire data is retrieved by a first query. Then, the retrieved data is passed to a subsequent query through a variable in the parent language. Individual subdata of the data passed by the subsequent query is separately processed. When this method is used, the subsequent query does not necessarily use all the subdata of the retrieved data. However, if the parallel database system technology described in Prior Art is applied intact to the ADT, all the data to returned as a result to the host computer is transferred to the front end server, which analyzes or executes the subsequent query, when retrieval or query is executed. When unused subdata is a large volume of data such as LOB data, the transfer time may be increased to take more time to execute a query if unused large volume of data is transferred from a database operation server to the front end server. An object of this invention is to minimize the query execution time by enabling only the data used for the subsequent query to be transferred from the database operation server to the front end server.

[Means for Solving the Problems]

To accomplish the above object, only the data positional information should be returned from the database operation server to the front end server when data comprising a plurality of pieces of subdata is used for a first retrieval query. The positional information includes an address of data on the database operation server, and an identifier of the database operation server. The positional information is passed from the front end server to a subsequent query through a variable. The subsequent query fetches each subdata of the data stored in the database operation server, depending on the passed positional information, the dictionary information on each subdata position in the data, and the identifier of the subdata required for the query. The positional information includes an identifier of a database operation server where data is stored, so a request of fetching the data can be issued to the database operation server where the data is stored. In addition, the positional information includes an address of data stored in the database operation server. Therefore the database operation server can fetch the data. By calculating the position of the required subdata from the data, each subdata can be fetched, depending on the dictionary information on each subdata position in the data and the identifier of the required subdata. The fetched

subdata is returned to the front end server, which enables a processing to be executed using the subdata. The database operation server transfers only the data to be used for a subsequent processing to the front end server, so an object of minimizing the query execution time may be accomplished if unused data volume is large.

When the subsequent processing with the subdata is a processing of updating the data obtained from the first retrieval query, the internal format processing procedure for updating on the database operation server side receives and updates the required subdata. In this case, no subdata is transferred from the database operation server to the front end server, so the query execution time may be further reduced.

Furthermore, it is assumed as a first method in which the above preceding retrieval query fetches only the data positional information and the subsequent query fetches the subdata. In addition, it is assumed as a second method in which the preceding retrieval query fetches the entire data including the subdata on the front end server side and passes the data to the subsequent query. One of the above two methods should be selected by comparing in the costs of the methods calculated from the subdata length or communication time, or by checking whether there is subdata longer than a standard value set

by the system. At the result, a method may be selected that enables a query execution time to be set shorter and properly according to the conditions, and the above object of minimizing the query execution time may be accomplished.

An explanation for this invention is provided as follows, focusing on a relational database and SQL, which are leading in the field of databases, as well as using the SQL3 ADT as the data with a complicated organization manipulated by the invention. However, the invention may be applied to a database management system that database queries are embedded in a parent language and data can be transmitted between a plurality of database queries, and that can manipulate data comprising a plurality of pieces of subdata.

[Mode for Carrying out the invention]

Referring to the drawings, an embodiment of this invention will be explained as follows.

Fig. 1 shows a configuration of an embodiment of this invention. This embodiment comprises a database server 12 operating as a front end server for a query 122 issued to a database, and a plurality of database operation servers 13 executing operation for databases.

It is supposed that the front end database server 12 should be connected to the database operation servers 13 through a high-speed mutual-connected network.

However, this invention may be applied to even a system that is comprises a single processor, not a parallel database system with a plurality of processors connected with each other through a network, and allocates parallel processes to individual servers.

The front end server 12 is supposed to be connected to an external host 11 through a network. However, this invention enables a data volume transmitted between the front end server 12 and the database operation servers 13 to be decreased. Therefore, the invention can be applied to an alternative idea of installing the host's functions in the database system and connecting to each other through an internal high-speed network, or an alternative idea of integrating the host's functions with the front end database server. In addition, the invention can be applied to even the case where a sequence of queries are the stored procedures, not a UAP in the host, because a plurality of pieces of subdata can be retrieved by a query, transmitted through a variable, or used for the subsequent query.

Fig. 1 is an example of analyzing or executing a database query. However, see Fig. 10 for details of analysis on table or type definition statements. It can be determined by semantics analysis whether a query is a database query or definition statement. An analysis 101 includes this determination.

The front end server 12 executes the analysis 101 for a first retrieval query 122a to create an internal format processing procedure 125. The internal format processing procedure may be an execution form code or a code for an interpreter.

After the retrieval query 122a is executed, it is considered that there is no manipulation for the subdata (132) and that the statement is a database operation statement (133), so the front end server 12 transfers the internal format processing procedure 125 to a database operation server 13 (102). The database operation server 13 receives the internal format processing procedure 125 (111), executes the internal format processing procedure 125, and obtains an identifier (ID) of the database operation server and an address of data when the data comprising a plurality of pieces of subdata is retrieved (112). The database operation server 13 transfers a retrieval result 126 to the front end server 12 (113). The front end server 12 receives the retrieval result 126 (103), and returns the received result to a UAP as information 127 to be passed to a subsequent query 122b through a variable (105).

Then, the front end server 12 executes an analysis (101) for the subsequent query 122b using the subdata. Because a variable that becomes an input exists in the host,

the information 127 bound to the variable is passed from the UAP to the front end server 12.

When the information 127 bound to the variable is an positional information 126 of data comprising a plurality of pieces of subdata, the front end server 12 obtains information 124 to fetch the subdata, from a dictionary 14. Using the above information 127 and 124, the front end server 12 creates an internal format processing procedure for obtaining subdata, and a procedure for processing a query using the obtained subdata.

Because there is a manipulation to the subdata for the query 122b executed (131), the front end server 12 obtains the subdata (106). Because a database operation server where the data is stored may be found from an identifier of the database operation server where the positional information 126 obtained by the first retrieval is stored, the front end server 12 sends a subdata fetching request together with information 128 required for fetching the subdata, to the database operation server 13. The information 128 required for fetching the subdata includes an address of data in the database operation server, and the offset information 124 for fetching the required subdata. The required subdata can be extracted when the front end server 12 executes the analysis (101), so the information should be embedded beforehand in an internal

format for obtaining the subdata. When receiving the subdata fetching request, the database operation server 13 fetches the subdata from the information in the data address and subdata offset position and returns subdata 129 to the requesting front end server 12. According to this embodiment, the processing by the database operation server 13 includes the receipt of the data address and subdata offset information, the return of the subdata, and the system allocation. However, this invention can be applied to an alternative idea that an internal format processing procedure for the above processing by the database operation server 13 should be created for its analysis by the front end server 12, and transferred to the database operation server 13 for its execution.

The front end server 12 receives the required subdata 129 (107). A location where the received subdata is stored should be pointed from a location of the subdata used in creating the internal format processing procedure for a processing using subdata primarily in the query 122b when the analysis (101) is executed. Not depending on whether the subsequent query is a database operation statement (133) or not (134), the subdata can be used in executing (104, 112) the internal format processing procedure.

Only the positional information 126 is obtained when

the retrieval query 122a is first executed for the data comprising a plurality of pieces of subdata, and only the required subdata 129 is obtained when the query 122b using the subdata is subsequently executed. Therefore, the query execution time may be minimized if the unused data volume is large same as the LOB.

Fig. 2 describes the examples of data comprising a plurality of pieces of subdata, i.e., an example of a data type definition 21, an example of type-created data 22, and an example of a data retrieval query 23.

The data type definition includes a name and a type declaration 201 of each subdata (i.e., attribute called "ADT"). A subdata type may be a system defined type or a user defined type. A function definition or procedure definition (202) for the data of the type, or a relationship in inheritance between the types may be specified or not.

Data comprising a plurality of pieces of subdata is used in a table definition 22, same as the system defined type. By inserting data into the table, data 203 comprising the subdata 204 including a zip code, an address, and a telephone number like address data can be created.

A query such as retrieval query can be executed for the created data (205, 206). A subdata fetching example 205 is a query for retrieving a zip code as subdata of the

address data, and a data fetching example is a query for retrieving address data that a subdata address of the address data is Yokohama. In this way, subdata as well as data comprising a plurality of pieces of subdata can be retrieved.

Fig. 3 shows an example of the positional information 126 in Fig. 1 according to this invention. The positional information 126 includes an identifier 301 of the database operation server 13 and an address 302 of the data in the database operation server 13. The information used for control processing by the front end serer 12 may include an additional information such as a type identifier. The identifier 301 of the database operation server 13 may be information that can specify a database operation server where the data is stored. The address 302 of the data may be represented by a real address where the data is stored, or a logical address specified by an offset from a first address whose data is fetched into a memory.

Fig. 4 shows an example of the information 127 to be bound to the variable in Fig. 1 according to this invention. Fig. 4 shows a case where the retrieval result of data comprising a plurality of pieces of subdata is bound to the variable. If the entire data comprising a plurality of pieces of subdata is not retrieved, the information to be bound to the variable may be the data itself to be

retrieved. The information 127 to be bound to the variable includes an identifier 401 of the database operation server 13 and an address 402 of the data in the database operation server 13.

The information used for control processing by the front end server 12 may include additional information such as a type identifier. The identifier 401 of the database operation server 13 may be information that can specify a database operation server where the data is stored. The address 402 of the data may be represented by a real address where the data is stored, or a logical address specified by an offset from a first address whose data is fetched into a memory. When a method is executed that selects whether data is stored in the front end server or the back end server, same as the embodiment in Fig. 21, information indicating whether the data is stored in the front end server or the back end server should be included the information to be bound to the variable. The UAP in the host needs to provide an area with a length enough to receive the information to be bound to the variable, as a variable area, when data comprising a plurality of pieces of subdata is retrieved.

Fig. 5 shows an example of the internal format processing procedure 125 in Fig. 1 according to this invention. The internal format processing procedure 125

comprises a code to be interpreted/executed by an interpreter and information added to each code. The information includes various types of information proper for a processing, for example, information on a type or length of data to be fetched, subdata offset information, and information on a location where fetched data is stored, as well as information on a code to be next executed. The code to be next executed can be branched, depending on a conditional branch information. Information 502 added to a subdata fetching code 501 includes the data positional information 126 and the offset information of the subdata to be used. This invention can be applied to the internal format processing procedure that is an execution form code, not a code to be interpreted/executed by an interpreter. Fig. 5 shows a relationship between an internal format processing procedure 51 for fetching the subdata 129 from the positional information 126, and an internal format processing procedure 52 using the subdata. Using the processing procedure 51, a location where the subdata 129 is stored should be represented as an offset position 506 on a shared memory, and contained in information 504 on a code using the subdata, using the processing procedure 52. Depending on the above information, the subdata 129 fetched by the processing procedure 51 can be used on the processing procedure 52 side. The processing procedure 51

may be integrated with the processing procedure 52. In the example of Fig. 5, a processing of fetching the subdata 129 is interpreted/executed by a code 501a, using the information 502 on the subdata to be fetched, but may comprise a plurality of codes representing the subdata fetching procedure. The example of Fig. 5 shows the internal format processing procedure 52 using the subdata 129. However, the processing procedure 51 for fetching the subdata and the information on the offset position in a location 505 where the subdata 129 is stored are not required for the internal format processing procedure without using the subdata. The information required for interpreting/executing the internal format processing procedure for fetching the positional information 126 includes the offset position of an area with a length of positional information to be fetched.

Fig. 6 shows an example of offset information 124 of the subdata in Fig. 1 according to this invention. The subdata offset information 124 should be created for each data comprising a plurality of pieces of subdata. The subdata offset information 124 includes a subdata identifier 601, a data type 602, a data length 603, and an offset position 604. The offset position is an offset from a starting address as a base address where data is stored. If each subdata is clustered, the data type 602 or the

offset position 604 may not be required because the offset position of each subdata can be calculated from the data length 603. If there is a variable-length subdata, the offset position cannot be set in the dictionary 14. In this case, an alternative idea of including the offset position in the data 130 can be applied. The offset position should be included in data 130 so that the subdata identifier 601 can correspond to the offset. Supposing the subdata identifiers are numbered in the ascending order, i.e., data definition order, beginning with "1," a method is considered that an offset is set in the data 130 in the same order.

Fig. 7 shows an example of the subdata fetching information 128 in Fig. 1 according to this invention. The subdata fetching information 128 includes a data address 302 in the database operation server 13, and information 701 of the subdata 129 to be used. The offset information 701 of the subdata 129 to be used includes the subdata identifier 601, the data type 602, the data length 603, and the offset position 604. For the offset information 701 of the subdata 129 to be used, the offset information of the identifier 601 of the subdata 129 to be used should be fetched from the subdata offset information 124 when the front end server 12 executes the analysis 101. An alternative idea can be also applied that the subdata

offset information 124 and the identifier of the subdata 129 to be used should be included in the subdata fetching information 128 and the offset information of the subdata 129 to be used should be selected on the database operation server 13 side.

Fig. 8 shows an example of the subdata 129 in Fig. 1 according to this invention. The subdata 129 is an actual data 801 of a system defined type or user defined type. When the subdata to be fetched is of a user defined type and the data to be used is subdata of the subdata, an alternative idea can be applied that the offset information of the subdata of the subdata should be included in the subdata fetching information 701 and the subdata of the subdata should be fetched on the database operation server 13 side. Even if the data to be used is subdata of the subdata of the subdata of.... (continuously repeated same words), the alternative idea can be applied by including the offset information in the subdata fetching information 701.

Figs. 9(a) and 9(b) show examples of the data 130 stored in the database operation server 13 in Fig. 1 according to this invention. The data 130 stored in the database operation server 13 is a set of data 901 in each column. To speed up a processing of fetching each column, additional information such as offset information from a

starting address may be prepared for each column data. Each column data includes a system defined type data 902 and a user defined type data 903.

Each column data type depends on a table definition by a CREATE TABLE definition statement, so both of the system defined type data 902 and the user defined type data 903 may appear zero or more times in any order. However, at least one of the above two data types is required. In Fig. 9, (a) is an example of data in a format that a column data comprising a plurality of pieces of subdata is clustered with other column data and the clustered data is embedded in the entire data 130. When the subdata includes a variable length data, as shown in Fig. 9 (b), an alternative idea can be applied that each subdata offset information 904 should be embedded in a column of data comprising a plurality of pieces of subdata. The format of the data 903b comprising a plurality of pieces of subdata is same as the format of the entire data 130. In Fig. 9 (b), data is embedded in offsets sequentially from a starting address where the subdata is stored, but other data format may be used if information with which the subdata position can be determined by the subdata identifier has been embedded. In addition, when a variable length subdata is used, an alternative idea is considered that information with the data length should be embedded in

the variable length subdata, not the offset information. In this case, this invention can be applied if the subdata can be fetched depending on the identifier of the subdata in the dictionary information in Fig. 6.

In addition, an alternative idea can be also applied that the column data 903 comprising a plurality of pieces of subdata should be stored in a different area from the entire data 130 and only the pointer to the area should be stored in the data 130.

Fig. 10 shows an embodiment of creating subdata offset information 124. The subdata offset information 124 is created when defining a type of data comprising a plurality of pieces of subdata. The front end server 12 executes the analysis 101 for an definition statement 1001 such as a CREATE TYPE definition statement.

The front end server 12 examines a type of each subdata, and calculates the subdata identifier 601, the data type 602, the data length 603, and the offset position 604, depending on a length determined by a type, or depending on a length defined when a character string is specified. The subdata identifier 601 may correspond to a subdata name. Discriminating the front end server from a server where a dictionary is installed, an alternative idea can be applied that an analysis should be executed on the server where the dictionary is installed.

Fig. 11 shows an embodiment of creating the data 130 comprising a plurality of pieces of subdata. The data 130 comprising a plurality of pieces of subdata is created when an insertion query is executed. A table definition information created by a table definition statement such as CREATE TABLE is used. The table definition information includes an identifier and a data type of each column. The current table definition information is different from the conventional table definition information in that a user defined type can be used as a data type. Fig. 11 is an example of the insertion query. The front end server 12 executes the analysis 101 for an insertion query 1101. The insertion query 1101 includes a numeric data to be inserted into each column. When the numeric data to be inserted into each column is data comprising a plurality of pieces of subdata, there are methods considered, e.g., a method of specifying each subdata value or of specifying a function and its argument, which create data (however, the argument may not be specified). When data is created by a function, a result of analyzing functions specified in the definition is stored in the dictionary when defining a type in Fig. 10.

By executing the analysis 101 for the insertion query 1101, the front end server 12 creates an internal format processing procedure 1102, the internal format of which includes a value to be inserted, a function and its

argument, which create a value, as information interpreted/executed by an interpreter. The information includes a type and length of each column or subdata obtained from the table definition information and subdata offset information. The front end server 12 transfers an internal format 1102 to the database operation server 13 where a table to be inserted is stored. The database operation server 13 receives and executes the internal format 1102. Based on the information on a type and length of each column or subdata and the information on a value to be inserted, a code executed by the interpreter creates the value to be inserted (1103), converts a type (1104), embeds the value to be inserted in the data 130 format, and stores the value in a database (1105). For the subdata of the subdata, the data 130 is embedded using a recursive processing. This invention may be applied even if the internal format processing procedure 1102 is an execution form code, not a code interpreted/executed by the interpreter.

Fig. 12 is a flowchart of creating the internal format processing procedure in Fig. 1 according to this invention. The front end server 12 receives the query 122, and the information 127 to be bound to a variable if data is input to the variable (1201). The front end server 12 executes a syntax analysis (1202) and a semantic analysis

(1203), during which the front end server 12 checks whether the variable subdata is used and analyzes the identifier of subdata to be used if the variable subdata is used. The query 122 may include a plurality of pieces of subdata, so the subdata to be used and the subdata to be fetched should be included with same identifiers in the analysis result information. If the variable subdata is used (1204), the front end server 12 fetches the data positional information 126 from the information 127 to be bound to the variable (1205), and creates the internal format processing procedure 51 for fetching the subdata from the data positional information 126, the offset information 124 of the subdata in the dictionary, and the identifier of the subdata to be used (1206). Then, the front end server 12 creates the internal format processing procedure 52 for the query 122 using the subdata (1207). By setting an offset indicating the same storage location in the subdata with same identifiers, the subdata can be transmitted between the subdata fetching side 51 and the subdata using side 52.

If the variable subdata is not used (1204), the subdata fetching internal format processing procedure is not required, so the front end server 12 creates only the internal format processing procedure for the query 122 (1207). When the query is executed for a database operation server in the database operation server side,

information on the database operation server to be executed should be added to the internal format processing procedure. The information on the database operation server to be executed is obtained from information on segmenting a table to be handled. The table segmentation information is specified when defining a table, and stored in the dictionary.

Fig. 13 is a flowchart showing an embodiment of transferring an internal format processing procedure (102) and receiving the processing procedure (111) according to this invention. The front end server fetches one of the identifiers of database operation servers 13 to be executed (1301), and transfers the internal format processing procedure 125 to the database operation server 13 (1302).

The database operation server 13 side receives the internal format processing procedure 125 (111), and executes the internal format processing procedure 125 (112). An alternative idea can be applied that the database operation server should return a report of receiving the processing procedure 125 to the front end server 12, send a start request to each of the database operation servers 13 after confirming that all of the processing procedure 125 has been received, and move control to the execution 112.

An improved idea can be also applied that, by storing the internal format processing procedure 125 in a

cache in the database operation server 13, the internal format processing procedure in the cache should be used for the second or subsequent executions according to an execution request issued.

Fig. 14 is a flowchart showing an embodiment of executing the internal format processing procedure 104 in Fig. 1 according to this invention. The interpreter fetches each code (1401) one by one and interprets/executes the code together with information added (1402). The interpreter fetches a code to be next executed, from the information (1403), and executes the subsequent codes one after another.

Fig. 15 is a flowchart showing an embodiment of executing the processing procedure (112) in Fig. 1 according to this invention. The interpreter executes the code, as shown in Fig. 14. There may be difference in the type of a code interpreted/executed by the interpreter between the front end server 12 side and the back end server 13 side. The interpreter fetches each code (1501) one by one and interprets/executes the code together with information added (1502). The interpreter fetches a code to be next executed, from the information (1503), and executes the subsequent codes one after another. When a code to be executed is used for retrieving data comprising a plurality of pieces of subdata (1504), the interpreter

obtains an identifier 301 of a database operation server 13a and an address 302 of the data, and creates the positional information 126 in an area prepared for storing an analysis result when the front end server 12 executes the analysis (101). The identifier of the database operation server 13 may be provided as information used by the internal format processing procedure 125. When data without a plurality of pieces of subdata is retrieved, the interpreter fetches/stores the data in the area provided for storing an analysis result when the front end server 12 executes the analysis (101).

Fig. 16 is a flowchart showing an embodiment of receiving the result (103) and an embodiment of transferring the result (113) in Fig. 1 according to this invention. The results are transferred from the database operation server 13, which has transferred the internal format processing procedure, to the front end server 12 (1601) until there are no execution results left in the database operation server 13 (1602). An alternative idea of transferring every two or more results can be applied. A result from each database operation server 13 is fetched from a queue in the order where the result is sent by the queue (1603). A processing procedure for receiving the result may be created when the front end server 12 executes the analysis (101). The front end server 12 continuously

receives the results (1603) until the database operation server 13 sends a report of "all results completely sent" to the front end server 12 (1604). The front end server 12 returns the results to the UAP (105). In Fig. 1, the front end server 12 returns all the received results, to the UAP, but in this case, an alternative idea can be applied that the front end server 12 should return one or more results to the UAP at each time of receipt. When the query is executed for retrieval, the result is a retrieval result. When data comprising a plurality of pieces of subdata is retrieved, the retrieval result includes the positional information 126. When the query is other than the retrieval result, a report of "all results completely sent" is merely issued.

Fig. 17 is a flowchart showing an embodiment of receiving the subdata (106), an embodiment of fetching the subdata (114), an embodiment of transferring the data (115), and an embodiment of receiving the data (107). The front end server 12 fetches the identifier of a database operation serer 13 where the data is stored, from the data positional information 127 (1701), sends a subdata fetching request to the database operation server 13 together with the information 128 for fetching the subdata (1702). The database operation server 13 side receives the information 128 for fetching the subdata, from the front end server 12

side (1703). The database operation server 13 obtains the data from the address 302 of the data in the database operation server 13 (1704). Using the offset information 701 of the subdata to be used, the database operation server 13 fetches the subdata 129 with the data length 603 from the offset position 604, depending on the data type 602 (1705). In the case of an alternative idea that an offset should be embedded beforehand in the data 130 for the variable length subdata, the offset embedded in the data 130 may be fetched from the subdata identifier and the subdata may be fetched using the offset. The database operation server 13 transfers the fetched subdata 129 to the front end server 12 (115). The front end server 12 stores one or more subdata 129 as results received (1706) from the database operation server 13, in the area 505 for storing the results (1707). This area 505 is specified by the offset 506 in the internal format processing procedure 52 using the subdata, so that the subdata can be used.

When the query 122b using the subdata enables the subdata to be transferred from the database operation server 13 side, same as the processing of updating the retrieved data, an alternative idea can be applied that a process of storing the subdata in an area shared with the side using the subdata (107) and the location 505 for storing the subdata should be set in the database operation

server 13 side. It can be determined whether the retrieved data is updated if an update retrieval is specified when the front end server 12 analyzes a first retrieval query 122a. In this case, there are no data transmissions between the front end server 12 and the database operation server 13, so that the query execution time is expected to be reduced.

When the query 122b using the subdata updates the retrieved data and the data address 402 is a real address for storing the data, an alternative idea can be applied that the data in a database should be updated by embedding the positional information 127 or the offset information 124 of the subdata to be used, directly in the internal format processing procedure for updating without fetching the subdata. In this case, the data is updated directly without being fetched or stored from/in a memory, so that the query execution time is expected to be decreased.

Figs. 18 and 19 outline the examples of applying this invention to the SQL. Fig. 18 shows an example of the SQL retrieving data comprising a plurality of pieces of subdata, and Fig. 19 is an example of the SQL using subdata of the data retrieved in Fig. 18. In the examples of Figs. 18 and 19, this invention can be applied to even the case where the front end server 12 fetches one retrieval result using an INTO, or fetches a plurality of retrieval results

and use the results one by one for the subsequent queries using a loop (which is an example to be used later).

In Fig. 18, the SQL for retrieving address data from an address book is analyzed/executed. A address data comprises three subdata, i.e., a zip code, an address, and a telephone number. The front end server 12 obtains/analyzes an address book table and a definition information on an address data type from the dictionary, and creates the internal format processing procedure 125. The address book table is assumed to be divided into two parts, which are separately stored in database operation servers 1 and 2, respectively. Data satisfying the WHERE condition is assumed to have been stored in the server 2. The front end server 12 transfers the internal format processing procedure to the servers 1 and 2 (102) and the servers 1 and 2 execute the internal format processing procedure (112). Data satisfying the WHERE condition has been stored in the server 2, so the server 2 obtains server 2 as an identifier, which is the identifier of the database operation server 13b (1505), obtains the address 1801 of the data (1506), creates the positional information 126 (1507), and returns the result 126 (113) to the front end server 12. The result 126 is returned as the information 127 to be bound to a variable, to the UAP side in the host 11. In Fig. 18, the disk is not shown, but an address may

be represented as a real address where data is stored, or as a logical address using an offset after fetching the data onto a memory.

Fig. 19 is an example of a query when a telephone number as subdata of address data is used as a determination condition. A processing after determination is not related to this invention, and therefore is omitted. The front end server 12 receives the query (1201), executes a syntax analysis and a semantic analysis, for which a telephone number is used as subdata of a variable ":X," and creates the internal format processing procedure 51 for fetching the subdata, depending on the positional information 127 bound to the variable and the subdata offset information 124 of the telephone number used as subdata (1206). Then, the front end server 12 creates the internal format processing procedure 52 of an IF statement using the subdata (1207). When the processing procedure 52 is executed, the front end server 12 transfers the address of the address data as subdata fetching information 128, and the offset information on the telephone number as subdata, to the server 2 as a database operation server where the data obtained from the positional information is stored. The offset of the telephone number is "26."

The zip code data length is "6," the address data length is "20," and the telephone number is stored in a

position of "26" from a starting address of the address data and can be fetched as subdata. However, the subdata is assumed to be clustered with each other. For simplified illustration, a starting address of the first subdata is represented as "0" in the offset. For a retrieval query in Fig. 18, an alternative idea can be applied that the address data should be stored in a cache and unnecessary I/O processing should not be executed. The front end server 12 stores the fetched subdata 129 in an area shared with an IF statement using the subdata when receiving the data (107) from the database operation server 13, and can use the subdata 129 for executing the IF statement processing procedure (104).

Fig. 20 shows an embodiment of an alternative idea according to this invention. Fig. 20 is an example of analyzing and executing the first retrieval query 122a. A difference from Fig. 1 is that the retrieval result 2001 is the data 2001 itself, not the positional information 126, when data comprising a plurality of pieces of subdata is retrieved (122a). In this case, the information 127 to be bound to a variable is the address of data in the front end server 12, or the data itself. If the UAP side has no function of receiving data comprising a plurality of pieces of subdata, the information 127 to be bound to a variable is the address of data in the front end server 12. When

the subsequent query 122b using the subdata is executed, the data is stored in the front end server 12 side, so the database operation server 13 side does not need to fetch the subdata. The database operation server 13 side can use directly the data or subdata in the front end server 12 side.

Fig. 21 shows another example of the information 127 to be bound to the variable in Fig. 1 according to this invention. The example of Fig. 21 is used in an example of Fig. 22. The information 127 to be bound to the variable includes a flag information 2101 indicating that the database operation server 13 has transferred the positional information 126 to the front end server 12 without transferring the data. If the database operation server 13 has transferred the data to the front end server 12 (flag bit=0), the flag information used when the database operation server 13 has transferred the positional information 126 to the front end server 12 (flag bit=1) may be used. If it can be determined whether the database operation server 13 has transferred the data to the front end server 12 or the positional information 126 to the front end server 12 without transferring the data, the above flag information may not be used. The information 127 to be bound to the variable includes information 2102 used when the database operation server 13 has transferred

the data to the front end server 12, and information 2103 used when the database operation server 13 has transferred the positional information 126 to the front end server 12. The information 2102 used when the database operation server 13 has transferred the data to the front end server 12 is the address of the data in the front end server 12, or the data itself. The information 2103 used when the database operation server 13 has transferred the positional information 126 to the front end server 12 is the positional information 126.

Figs. 22 and 23 outline the flows showing the embodiments of selecting the method of Fig. 1 (i.e., method of transferring the positional information) or the method of Fig. 20 (i.e., method of transferring the data) based on the selection criteria such as cost calculation. The selection criteria include a method of calculating/comparing the costs of various transfer methods depending on the dictionary information such as the subdata length, and a method of selecting a method 2202 of Fig. 1 if a large volume of subdata such as LOB data is included in the data to be retrieved or otherwise selecting a method 2203 of Fig. 20.

Fig. 22 outlines a flow of retrieving data comprising a plurality of pieces of subdata. The front end server 12 side selects the analysis/execution of the method

2202 of Fig. 1 or the method 2203 of Fig. 20 when executing the costs of various transfer methods. However, the front end server 12 side creates the information 127 to be bound to the variable in Fig. 21 when returning the result (105). The UAP side in the host needs to prepare an area with a length enough to receive the information to be bound to the variable, as a variable area, when the data including the subdata is retrieved.

Fig. 23 outlines a flow of executing a query using subdata. When executing the analysis (101), the front end server 12 selects the analysis/execution of 2301, the method 2302 of Fig. 1 or the method 2303 of Fig. 20, depending on the flag information 2101 indicating that the data of the information 127 to be bound to the variable is transferred to the front end server 12 or that the positional information 126 is transferred to the front end server 12 without the data being transferred.

[Effect of the Invention]

As explained above, according to this invention, a database operation server transfers only a positional information to a front end server for retrieving data comprising a plurality of pieces of subdata and fetches subdata to be used for a query using the subdata of the positional information, so that a communication time of transferring unused subdata can be reduced and a query

execution time can be minimized. In addition, the invention is effective more particularly for a large volume of data such as LOB data.

[Brief Description of the Drawings]

[Fig. 1]

Fig. 1 shows a configuration of an embodiment of this invention.

[Fig. 2]

Fig. 2 describes the examples of data comprising a plurality of pieces of subdata.

[Fig. 3]

Fig. 3 shows an example of the positional information.

[Fig. 4]

Fig. 4 shows an example of the information to be bound to the variable.

[Fig. 5]

Fig. 5 shows an example of the internal format processing procedure.

[Fig. 6]

Fig. 6 shows an example of subdata offset information.

[Fig. 7]

Fig. 7 shows an example of the subdata fetching

information.

[Fig. 8]

Fig. 8 shows an example of the subdata.

[Fig. 9]

Figs. 9(a) and 9(b) show examples of the data stored in the database operation server.

[Fig. 10]

Fig. 10 shows an embodiment of creating subdata offset information.

[Fig. 11]

Fig. 11 shows an embodiment of creating the data comprising a plurality of pieces of subdata.

[Fig. 12]

Fig. 12 is a flowchart of creating the internal format processing procedure.

[Fig. 13]

Fig. 13 is a flowchart showing an embodiment of transferring an internal format processing procedure and receiving the processing procedure.

[Fig. 14]

Fig. 14 is a flowchart showing an embodiment of executing the internal format processing procedure.

[Fig. 15]

Fig. 15 is a flowchart showing an embodiment of executing the processing procedure.

[Fig. 16]

Fig. 16 is a flowchart showing an embodiment of receiving and transferring the result.

[Fig. 17]

Fig. 17 is a flowchart showing an embodiment of obtaining and fetching the subdata.

[Fig. 18]

Fig. 18 outlines a sample.

[Fig. 19]

Fig. 19 outlines a sample.

[Fig. 20]

Fig. 20 shows an embodiment of an alternative idea.

[Fig. 21]

Fig. 21 shows an example of the information to be bound to a variable.

[Fig. 22]

Fig. 22 outlines a flow of retrieving data comprising a plurality of pieces of subdata.

[Fig. 23]

Fig. 23 outlines a flow of executing a query using subdata.

[Explanation of Reference Numerals]

11 … Host, 12 … Front end server, 13 … Database operation server, 14 … Dictionary, 101 … Analysis, 102 … Transfer the internal format processing procedure, 103 …

Receive a result, 104 … Execute the internal format processing procedure, 106 … Obtain subdata, 107 … Receive the data, 111 … Receive the processing procedure, 112 … Execute the processing procedure, 113 … Transfer the result, 114 … Fetch the subdata, 115 … Transfer the data, 121 … UAP, 122 … Query, 124 … Subdata offset information, 125 … Processing procedure, 126 … Positional information, 127 … Information to be bound to a variable, 128 … Subdata fetching information, 129 … Subdata, 130 … Data

[Name of Document] Abstract

[Summary]

[Subject] To minimize a query execution time when there is a large volume of unused data so as to transfer only data to be used for a subsequent processing from a database operation server to a front end server in a parallel database system.

[Solution] A parallel database system comprising a server which operates as a front end server for a query issued to a database, and a plurality of servers which operate databases as database operation servers, wherein the front end server is connected to the database operation servers through a network. When a first retrieval query is executed, a database operation server returns only the data positional information to the front end server. When a subsequent query is executed, the database operation server passes the data positional information to the front end server through a variable. For the subsequent query, the front end server fetches subdata, depending on the passed data positional information and the dictionary information of each subdata position in the data.

[Selected Drawing] Fig. 1

Drawings

[Fig. 1]

89 Analysis

90 Execution

91 Is the subdata manipulated?

92 Is this a database operation statement?

93 Disk

11 Host

12 Front end server

13 Database operation server group

14 Dictionary

101 Analysis

Create an internal format processing procedure

102 Transfer the internal format processing procedure

103 Receive a result

104 Execute the internal format processing procedure

105 Return the result

106 Obtain subdata

(Transfer an address of the data and offset information of subdata to be used, to a database operation server indicated by the data positional information (i.e., database operation server ID and data address))

107 Receive the data

111 Receive the processing procedure

112 Execute the processing procedure

(When data comprising a plurality of pieces of subdata is retrieved by a retrieval query, obtain the database operation server ID and the data address as the positional information)

113 Transfer the result

114 Fetch the subdata.

(Fetch the subdata from the data address and offset information)

115 Transfer the data

122a Query i

122b Query j

123 Control, process

124 Definition information

Subdata offset information

125 Processing procedure

126 Positional information

127 Information to be bound to a variable

128 Subdata fetching information

129 Subdata

130 Data

[Fig. 2]

94 CREATE TYPE address type

95 PUBLIC zip code CHAR (6)

96 PUBLIC address CHAR (20)

97 PUBLIC telephone number CHAR (10)

98 PUBLIC FUNCTION address type (number int)

99 RETURNS address type

910 Function body

911 CREATE TABLE address book

912 (id INT, name CHAR(10), address data address type);

913 Table Address Book

914 Name

915 Address data

916 Zip code

917 Address

918 Telephone number

919 SELECT zip code (address data) FROM address book

920 SELECT address data FROM address book

921 WHERE address (address data) = Yokohama

21 Example of type definition of data comprising a plurality of pieces of subdata

22 Example of data to be created from the type

23 Example of a retrieval query of the data comprising a plurality of pieces of subdata

201 Subdata definition

202 Function definition

203 Example of data comprising a plurality of pieces of subdata

204 Example of subdata

205 Example of fetching subdata

206 Example of fetching data

[Fig. 3]

126 Positional information

301 ID of the database operation server

302 Address of the data in the database operation server

[Fig. 4]

127 Information to be bound to a variable

401 ID of the database operation server

402 Address of the data in the database operation server

[Fig. 5]

922 Code

923 Information

51 Internal format processing procedure for fetching subdata

52 Internal format processing procedure using subdata

501 Subdata fetching code

502 Positional information

Subdata offset information

Offset list of a location where subdata is stored

Offset of a code to be next executed

503a Code using subdata 1

503b Code using subdata 2

504a Information required for processing

Offset of a location where the subdata 1 is stored

Offset of a code to be next executed

504b Information required for processing

Offset of a location where the subdata 2 is stored

Offset of a code to be next executed

505a A location where the subdata 1 is stored

505b A location where the subdata 2 is stored

506a Offset of a location where the subdata 1 is stored

506b Offset of a location where the subdata 2 is stored

507 Offset list of a location where subdata is stored

[Fig. 6]

924 Subdata 1

925 Subdata 2

926 Subdata 3

124 Subdata offset information

601 Identifier

602 Data type

603 Data length

604 Offset

[Fig. 7]

927 When subdata 3 is fetched

128 Subdata fetching information

302 Address of the data in the database operation server

601 Subdata 3

701 Offset information list of the subdata to be used

[Fig. 8]

928 Example of the subdata 3 in Fig. 7
(Actually in the FLOAT format)

129 Subdata 3

801 Actual data

[Fig. 9]

929 Column number

930 Offset information of each column

931 (a) Example of data stored in a database operation server

932 Number of subdata

933 (b) Example of data comprising a variable length subdata

130a Data stored in a database operation server

130b Data stored in a database operation server

901a Column 1

901b Column 2

903a Data comprising a plurality of pieces of subdata

901c Column 3

901d Column 1

901e Column 2

903b Data comprising a plurality of pieces of subdata

901f Column 3

[Fig. 10]

934 Syntax analysis

935 Subdata analysis

936 Analyze the next subdata

937 Is there the next subdata?

938 Analyze the inheritance and functions

939 Report the end of analysis

11 Host

12 Front end server

14 Dictionary

124 Definition information

Offset information of subdata

Information on the inheritance and functions

1001 Type definition statement

[Fig. 11]

940 Analysis

941 Execution

942 Outline of the processing procedure

943 Code set

944 Disk

945 Control, process

11 Host

12 Front end server

13 Database operation server group

14 Dictionary

101 Analysis

Create an internal format processing procedure

102 Transfer the internal format processing procedure

103 Receive a result

105 Return the result

111 Receive the processing procedure

112 Execute the processing procedure.

113 Transfer the result

124 Definition information

Subdata offset information

130 Data

1101 Insertion query

1102 Processing procedure

1103 Create a value to be inserted

1104 Convert the type

1105 Store the data

[Fig. 12]

946 Dictionary

124 Subdata offset information

1201 Receive the next query

1202 Syntax analysis

1203 Semantic analysis

1204 Is variable subdata used?

1205 Fetch the positional information bound to the variable

1206 Create an internal format for fetching the subdata

1207 Create an internal format for query

[Fig. 13]

947 Front end server

948 Database operation server side

949 Are there servers to be executed?

111 Receive the internal format processing procedure

1301 Fetch an identifier of a database operation server to be executed

1302 Transfer an internal format processing procedure to a fetched database operation server

[Fig. 14]

1401 Fetch a code to be next executed

1402 Fetch the information and execute the code

1403 Are there codes to be executed?

[Fig. 15]

1501 Fetch a code to be next executed

1503 Are there codes to be executed?

1504 Is this a retrieval of data comprising a plurality of subdata?

1505 Obtain an identifier of a database operation server

1506 Obtain the data address

1507 Create the positional information in an area for storing a result

[Fig. 16]

951 Front end server side

952 Database operation server side

953 A report of "all results completely sent" is issued?

954 Transfer the report of "all results completely sent"

105 Return the result to the UAP

1601 Transfer the result

1602 Are there results to be transferred?

1603 Fetch one result from the queue

1604 Have all executed servers issued a report of "all results completely sent"?

[Fig. 17]

115 Transfer the subdata to the front end server

1701 Fetch the identifier of a database operation server

1702 Send subdata fetching request to the fetched server together with the required information

1703 Receive the subdata fetching request together with the required information

1704 Obtain the data from the data address

1705 Fetch the subdata from the offset information of the subdata to be used

1706 Receive the subdata

1707 Store the subdata in an area shared with the side using the subdata

[Fig. 18]

955 Query i

956 Receive the query

```
SELECT address data INTO:X
FROM address book WHERE name = 'Sato'
```

957 Analysis

958 Definition information

959 Bind the result to :X

960 Syntax analysis

Semantic analysis

Create an internal format processing procedure

961 Server 2

962 Starting address

963 Execution

964 Address data

965 Zip code

966 Address

967 Telephone number

11 Host

12 Front end server

13 Database operation server group

13a Server 1

13b Server 2

14 Dictionary

102 Transfer the internal format processing procedure

103 Receive the result

105 Return the result

125 Processing procedure

126 Positional information

127 Information to be bound to a variable

1801 Starting address

[Fig. 19]

968 Query j

969 Receive the query

970 IF(zip code(:X)=224) THEN...

971 Analysis

972 Server 2

973 Starting address of the address data

974 Subdata offset information

975 Zip code

976 Address

977 Telephone number

978 Execution

979 Address data

980 Data length

11 Host

12 Front end server

13 Database operation server group

13a Server 1

13b Server 2

14 Dictionary

104 Execute a procedure of processing an IF statement or subsequent statements using the subdata

105 Return the result

106 Transfer the data address and offset information to

server 2

107 Receive the data

128 Starting address of the address data

Offset information of the telephone number

129 Telephone number

1206 Create subdata fetching processing procedure

1207 Create a processing procedure using the subdata

[Fig. 20]

981 Control, process

982 Analysis

983 Execution

984 Zip code

985 Address

986 Telephone number

987 Disk

988 Is this a database operation statement?

11 Host

12 Front end server

13 Database operation server group

14 Dictionary

101 Analysis

Create an internal format processing procedure

102 Transfer the internal format processing procedure

103 Receive a result

104 Execute the internal format processing procedure

105 Return the result

111 Receive the processing procedure

112 Execute the processing procedure.
(When data comprising a plurality of pieces of subdata is retrieved, obtain all subdata as a result)

113 Transfer the result

121 UAP

122a Query i

124 Definition information
Subdata offset information

125 Processing procedure

127 Information to be bound to a variable

130 Data

2001 Result

127 Information to be bound to a variable

2101 Flag indicating data transfer or positional information transfer

2102 Information when data transfer is executed

2103 Information when positional information transfer is executed

[Fig. 22]

989 Front end server side

990 Analysis

991 Is the cost of the method of transferring the positional information proper?

1201 Receive the next query

1202 Syntax analysis

1203 Semantic analysis

2201 Cost calculation

2202 Method of Fig. 1

2203 Method of Fig. 20

[Fig. 23]

992 Front end server side

993 Analysis

994 Has the positional information been transferred?

1201 Receive the next query

1202 Syntax analysis

1203 Semantic analysis

2301 Fetch the flag information

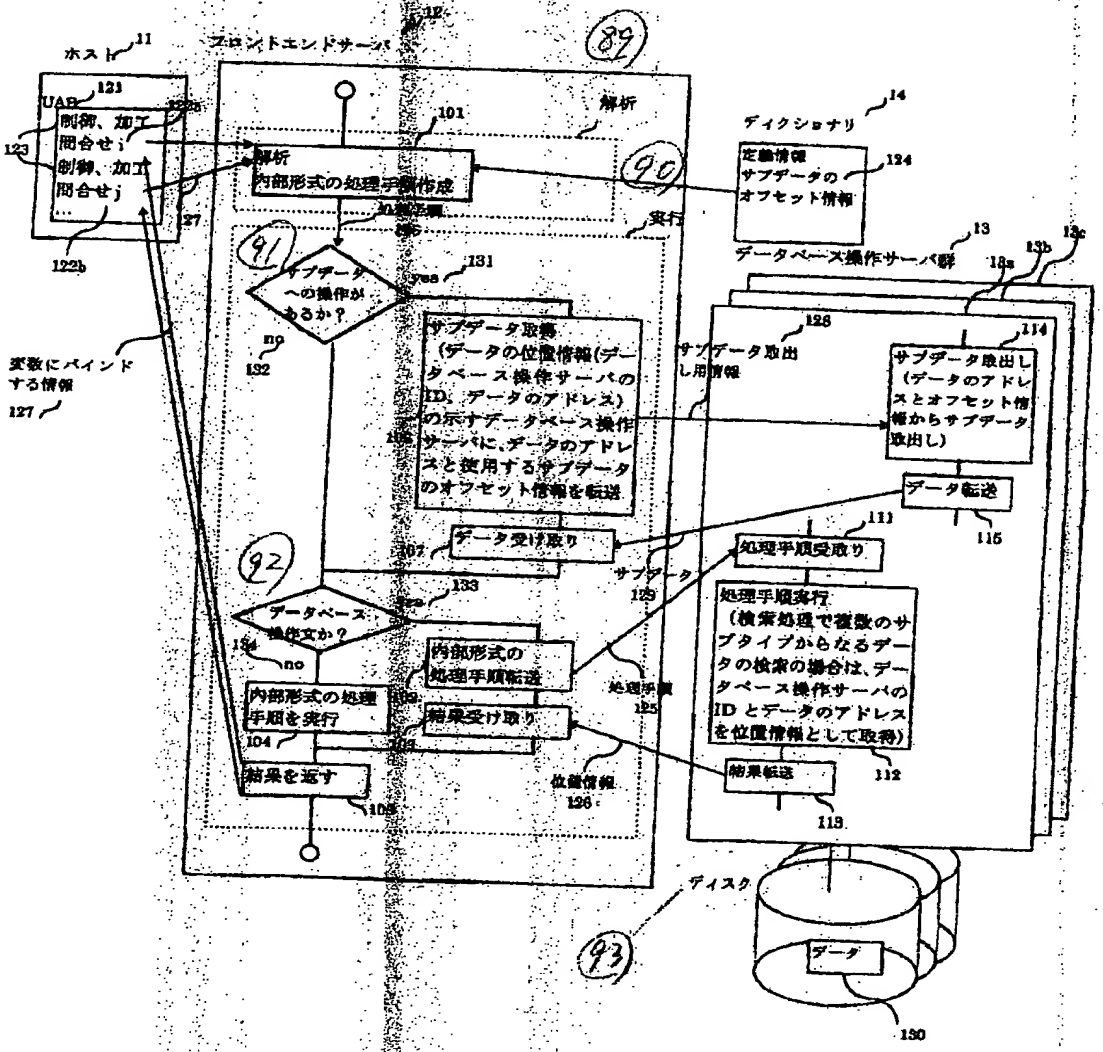
2302 Method of Fig. 1

2303 Method of Fig. 20

【書類名】　図面

【図１】

図１



【図２】

図２

```
複数のサブデータからなるデータの型の定義例                          21
CREATE TYPE 住所型 (
  PUBLIC 郵便番号 CHAR(6),
  PUBLIC 住所 CHAR(20),                                  } サブデータの定義  201
  PUBLIC 電話番号 CHAR(10),
  PUBLIC FUNCTION 住所型 (number int)
  RETURNS 住所型                                         } 関数の定義  202
  関数本体
  RETURN address_data) ;

型から作成されるデータの例                                      22
CREATE TABLE 住所録
 (id INT, 名前 CHAR(10), 住所データ 住所型) ;
```

表 住所録　　サブデータの例　204

id	名前	住所データ
1	Yamada	郵便番号：114 住所：Tokyo 電話番号：03-3735-XXXX
2	Sato	郵便番号：223 住所：Yokohama 電話番号：045-564-XXXX
3	Tanaka	郵便番号：830 住所：Hikata 電話番号：0942-43-XXXX

203 サブデータからなるデータの例

```
複数のサブデータからなるデータの検索問い合わせ例                  23
SELECT 郵便番号(住所データ) FROM 住所録 ;        205 サブデータを取出す例

SELECT 住所データ FROM 住所録
WHERE 住所（住所データ） = Yokohama ;        206 データを取出す例
```

【図３】

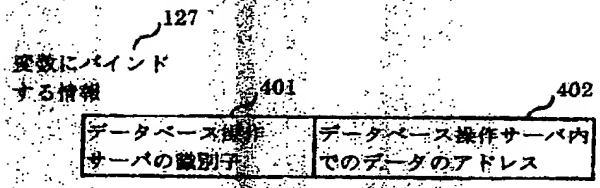
図３　　１２６

126 位置情報

301 データベース操作サーバの識別子	302 データベース操作サーバ内でのデータのアドレス

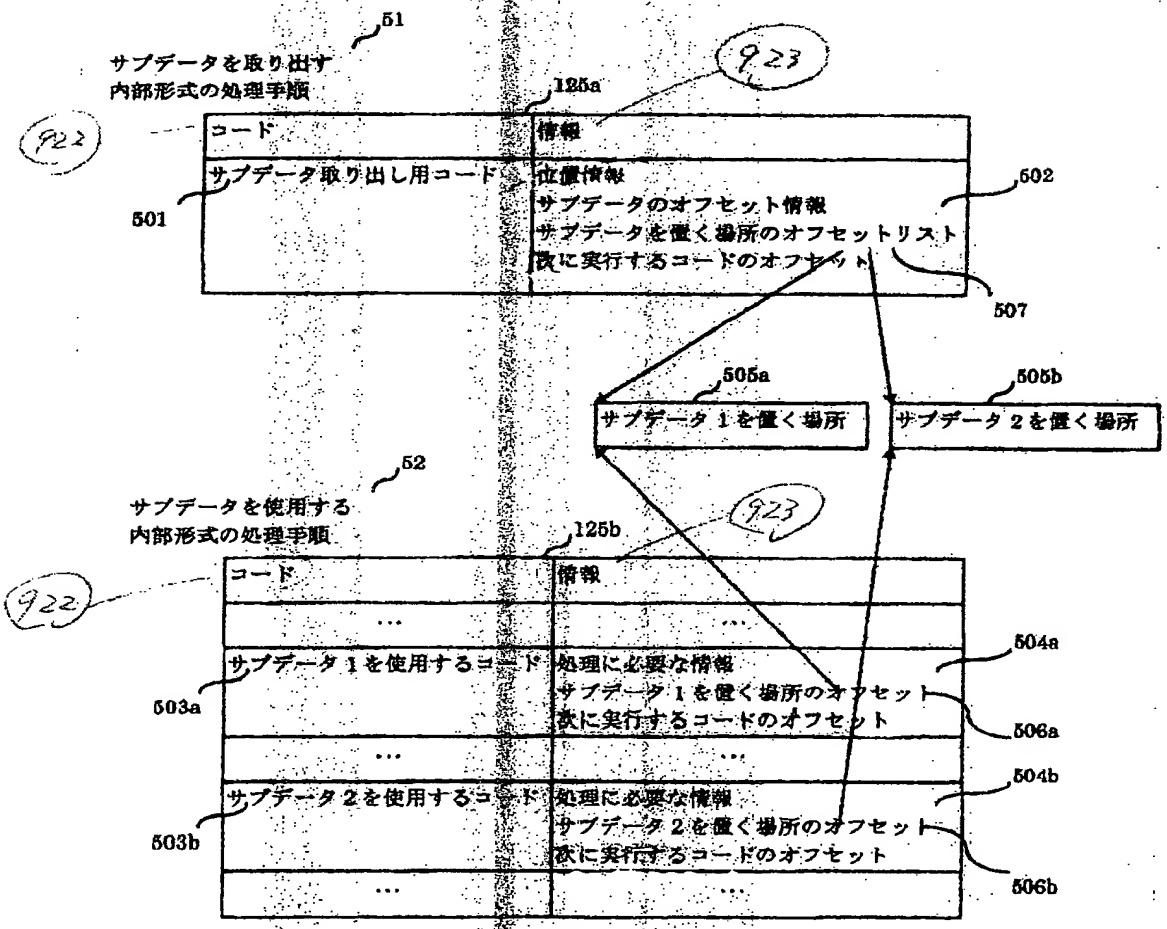
【図４】

図４　　１２７



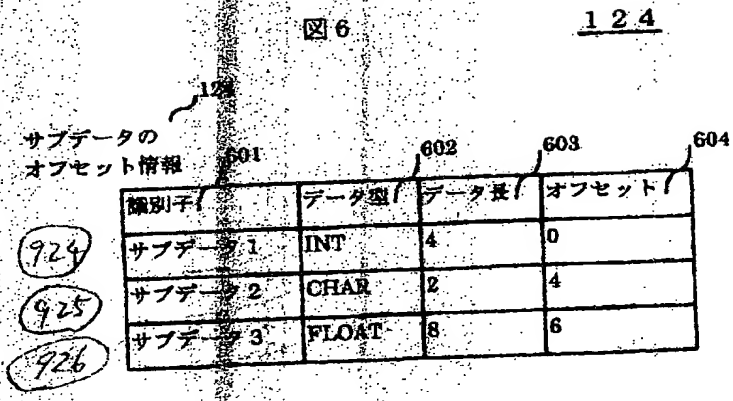
【図５】

図５　　１２５



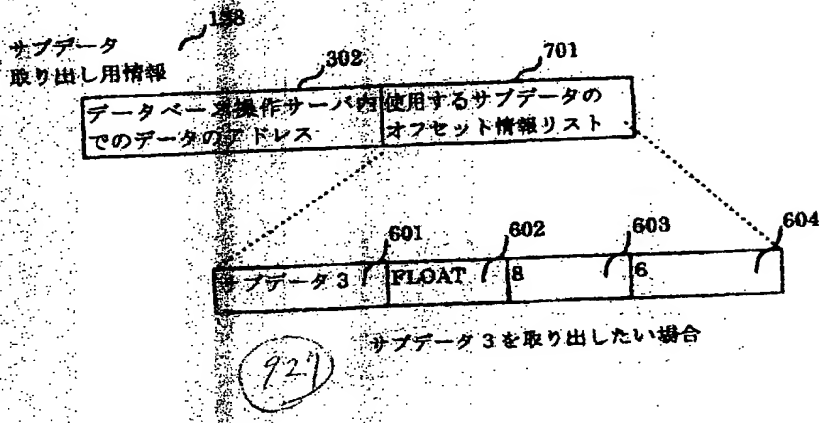
【図６】

図６


124
サブデータの
オフセット情報
601
602
603
604
識別子
データ型
データ長
オフセット
924
サブデータ１
INT
4
0
925
サブデータ２
CHAR
2
4
926
サブデータ３
FLOAT
8
6

【図７】

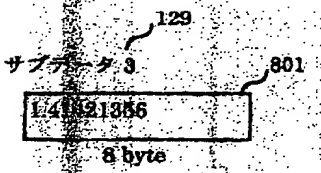
図７


128
サブデータ
取り出し用情報
302
701
データベース操作サーバ内
でのデータのアドレス
使用するサブデータの
オフセット情報リスト
601
602
603
604
サブデータ３
FLOAT
8
6
サブデータ３を取り出したい場合
927

【図８】

図8

129

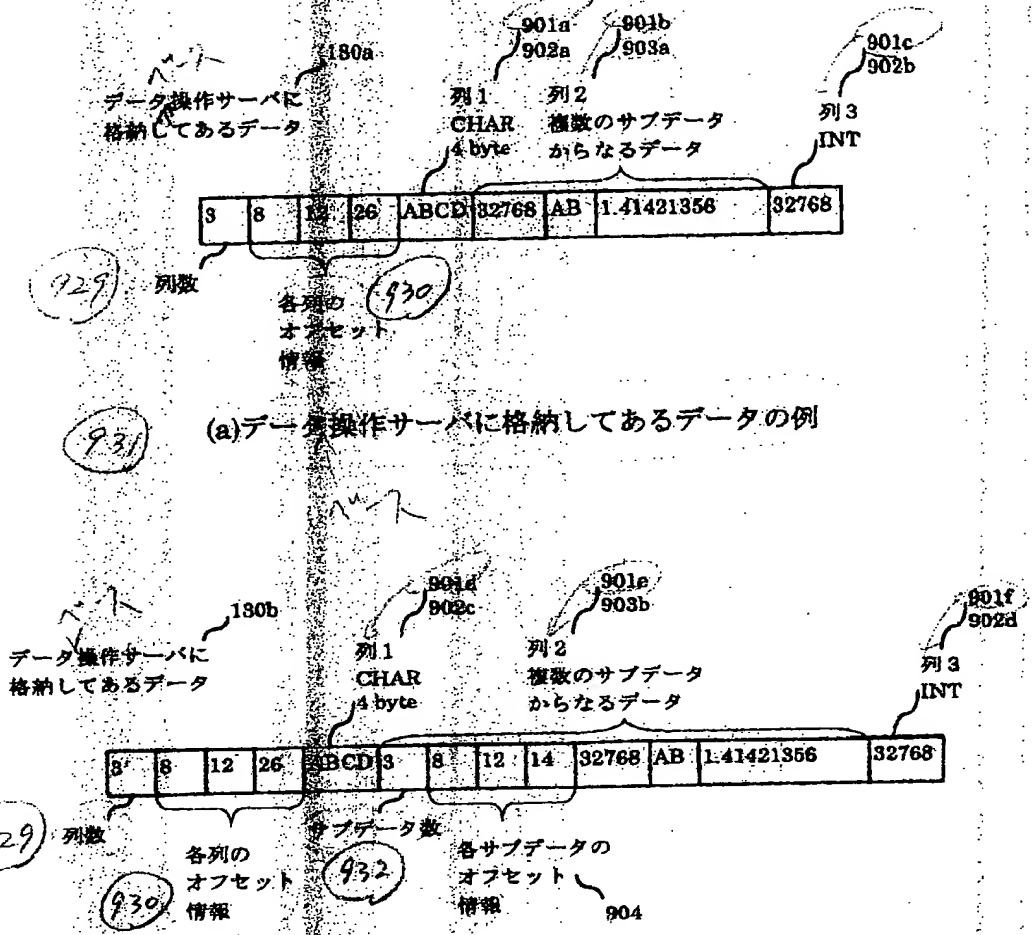


928

図7のサブデータ3の例
(実数は FLOAT の形式)

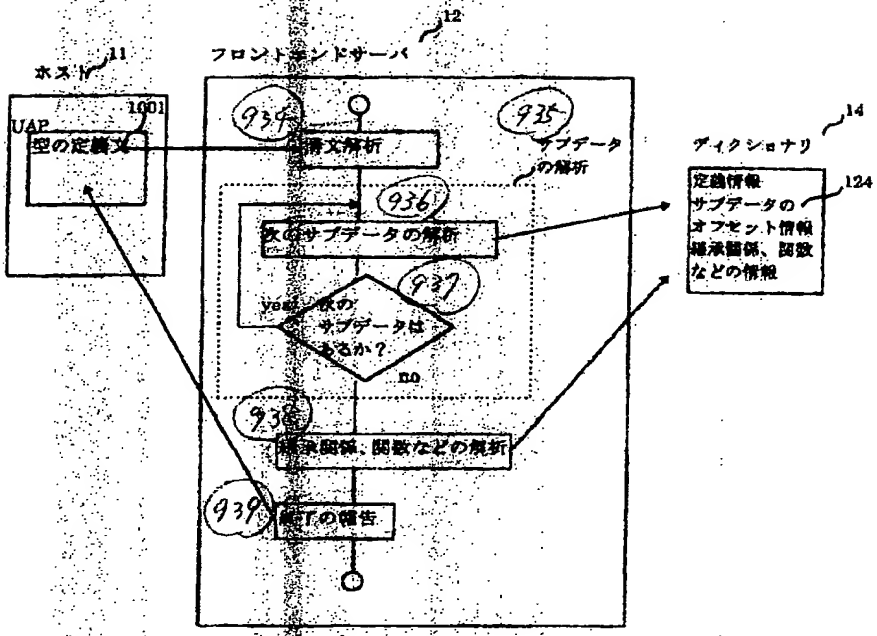
【図9】

図９


１３０
180a
901a
902a
901b
903a
901c
902b
データ操作サーバに格納してあるデータ
列１ CHAR 4 byte
列２ 複数のサブデータからなるデータ
列３ INT
3 8 12 26 ABCD 32768 AB 1.41421356 32768
929
列数
930
各列のオフセット情報
931
(a)データ操作サーバに格納してあるデータの例
180b
901d
902c
901e
903b
901f
902d
データ操作サーバに格納してあるデータ
列１ CHAR 4 byte
列２ 複数のサブデータからなるデータ
列３ INT
3 8 12 26 ABCD 3 8 12 14 32768 AB 1.41421356 32768
929
列数
930
各列のオフセット情報
932
サブデータ数
各サブデータのオフセット情報
904
933
(b)可変長のサブデータがあるデータの例

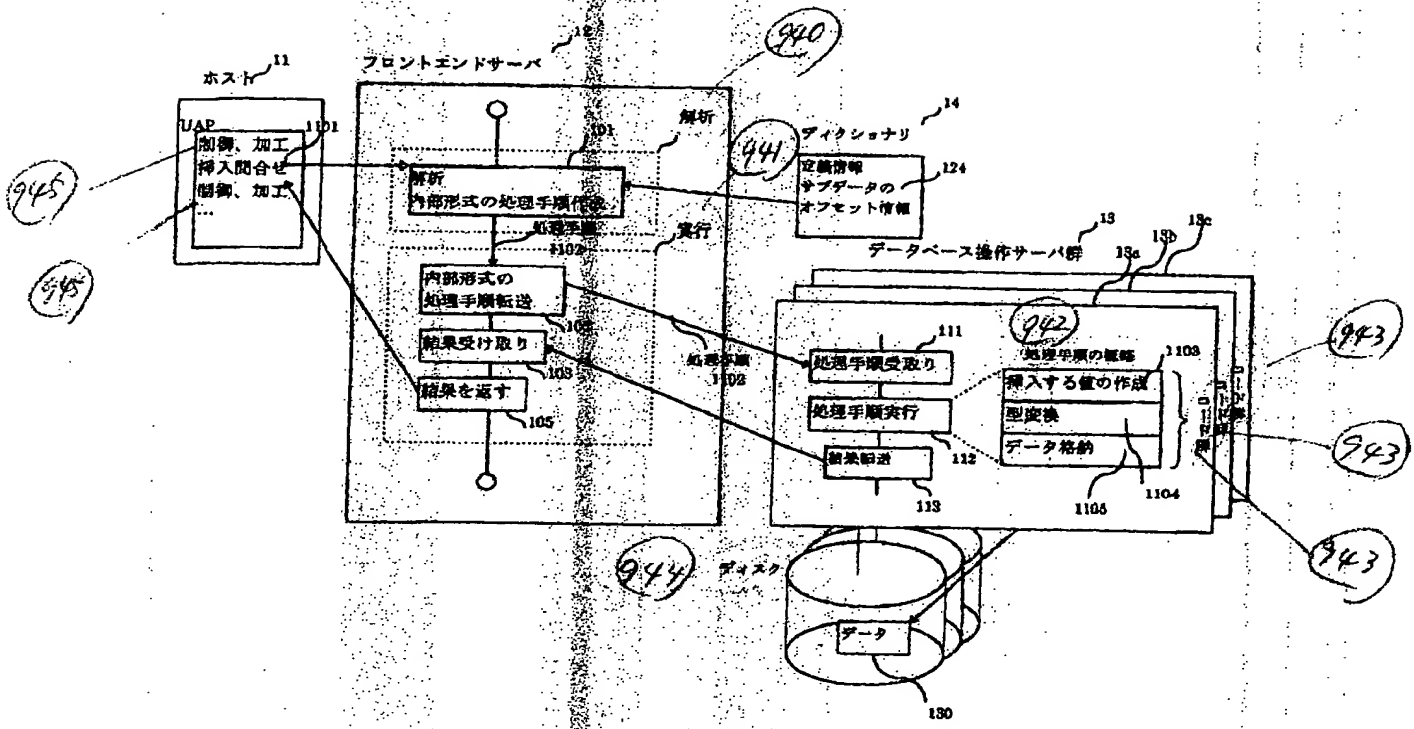
【図１０】

図１０


ホスト
11
UAP
1001
型の定義文
フロントエンドサーバ
12
935
サブデータの解析
936
次のサブデータの解析
937
yes
次のサブデータはあるか？
no
938
939
ディクショナリ
14
定義情報
サブデータのオフセット情報
継承関係、関数などの情報
124

【図１１】

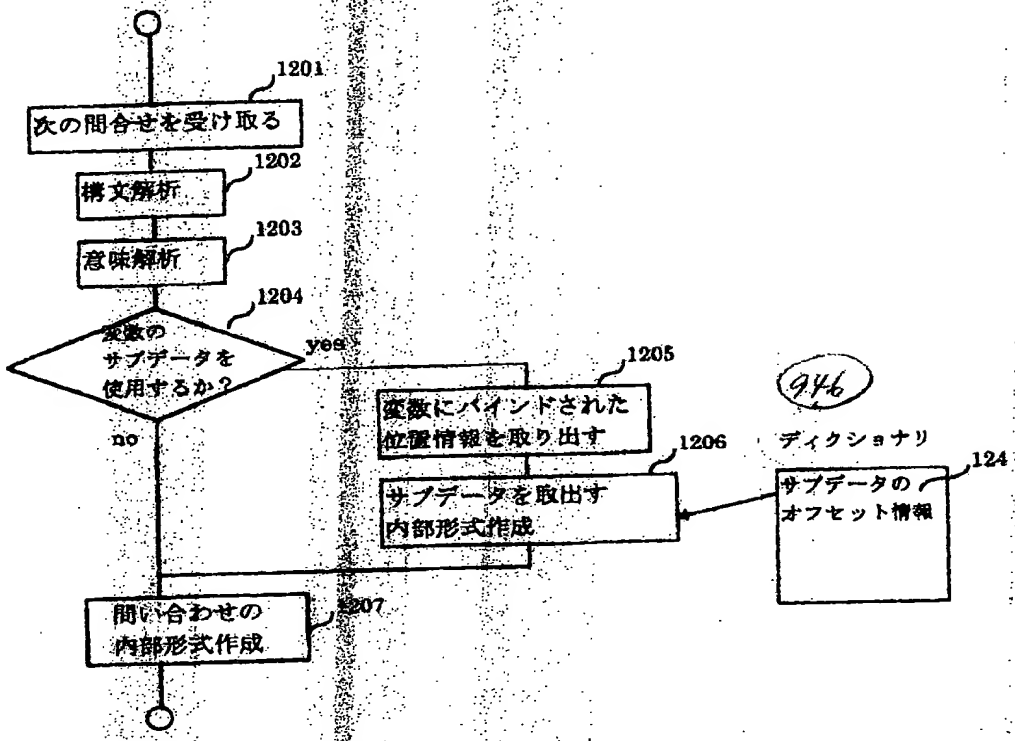
図１１


ホスト
11
フロントエンドサーバ
12
940
UAP
1101
制御、加工
挿入問合せ
制御、加工
...
945
945
解析
101
解析
内部形式の処理手順作成
941
ディクショナリ
14
定義情報
サブデータの
オフセット情報
124
実行
処理手順
1102
内部形式の
処理手順転送
102
結果受け取り
103
結果を返す
105
処理手順
1102
データベース操作サーバ群
13
13a
13b
13c
942
111
処理手順受取り
処理手順実行
112
結果転送
113
処理手順の概略
1103
挿入する値の作成
型変換
データ格納
1105
1104
コード部
943
943
944
ディスク
データ
130

【図１２】

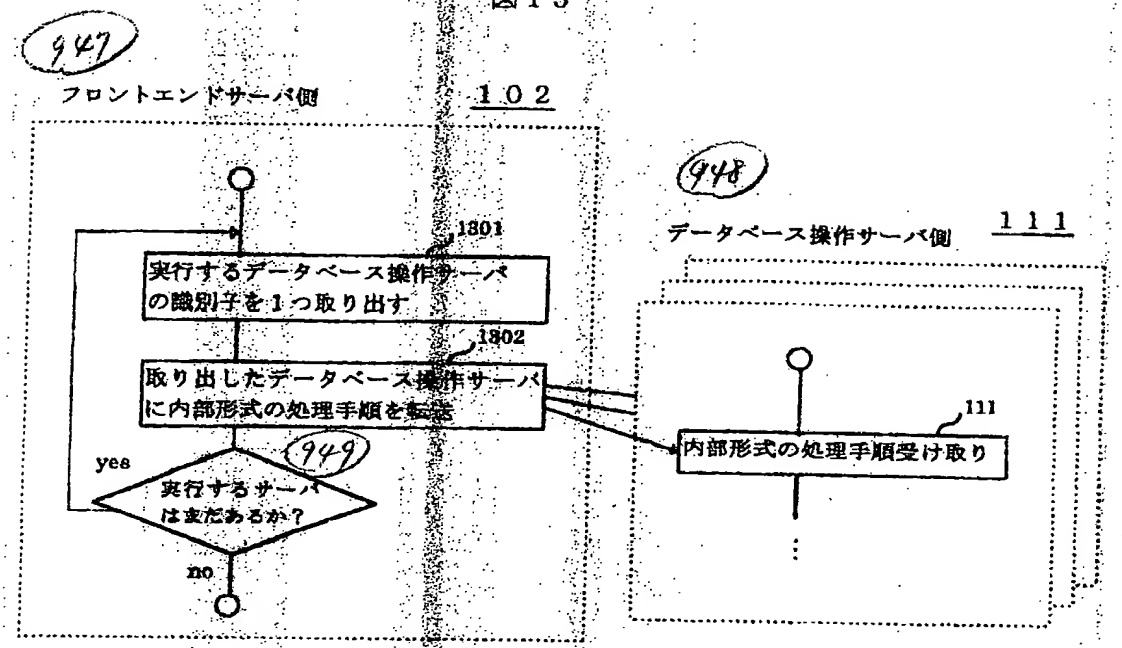
図１２

１０１


1201
次の問合せを受け取る
1202
構文解析
1203
意味解析
1204
変数の
サブデータを
使用するか？
yes
no
1205
変数にバインドされた
位置情報を取り出す
1206
サブデータを取出す
内部形式作成
946
ディクショナリ
124
サブデータの
オフセット情報
1207
問い合わせの
内部形式作成

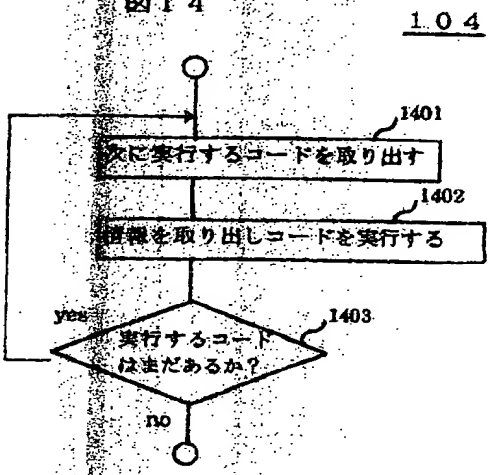
【図１３】

図１３



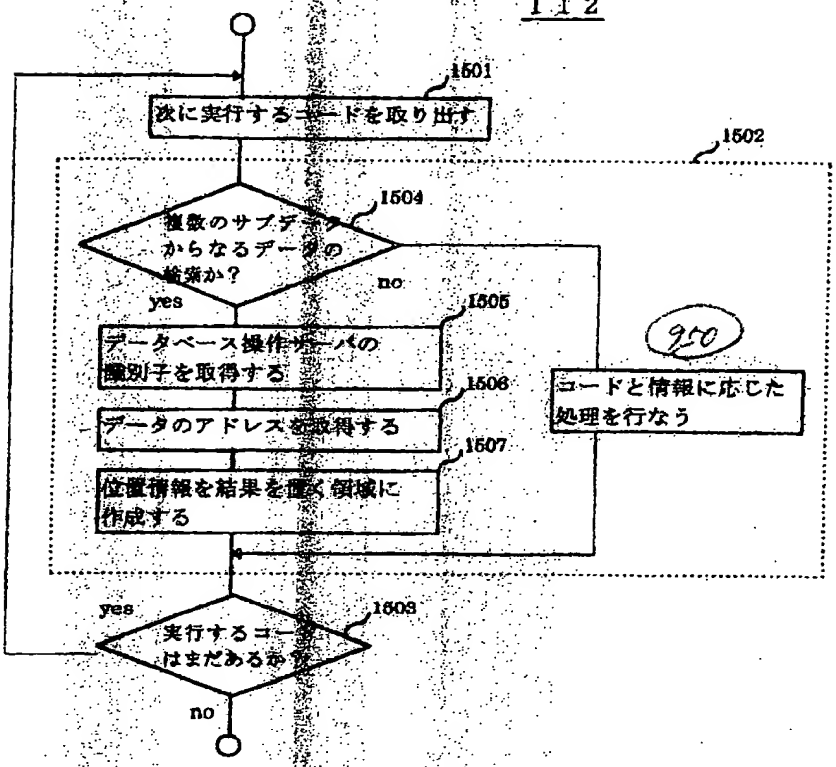
【図１４】

図１４



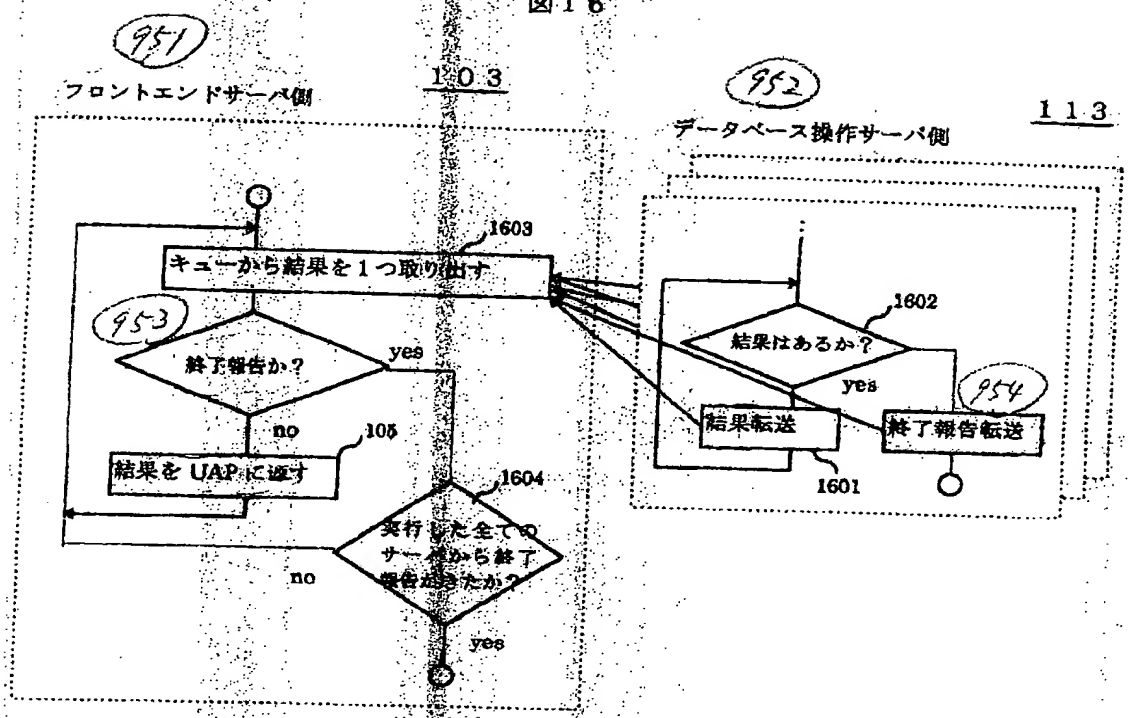
【図１５】

図１５


112
1501
次に実行するコードを取り出す
1502
1504
複数のサブデータからなるデータの検索か？
yes
no
1505
データベース操作サーバの識別子を取得する
1506
データのアドレスを取得する
1507
位置情報を結果を置く領域に作成する
950
コードと情報に応じた処理を行なう
yes
1503
実行するコードはまだあるか？
no

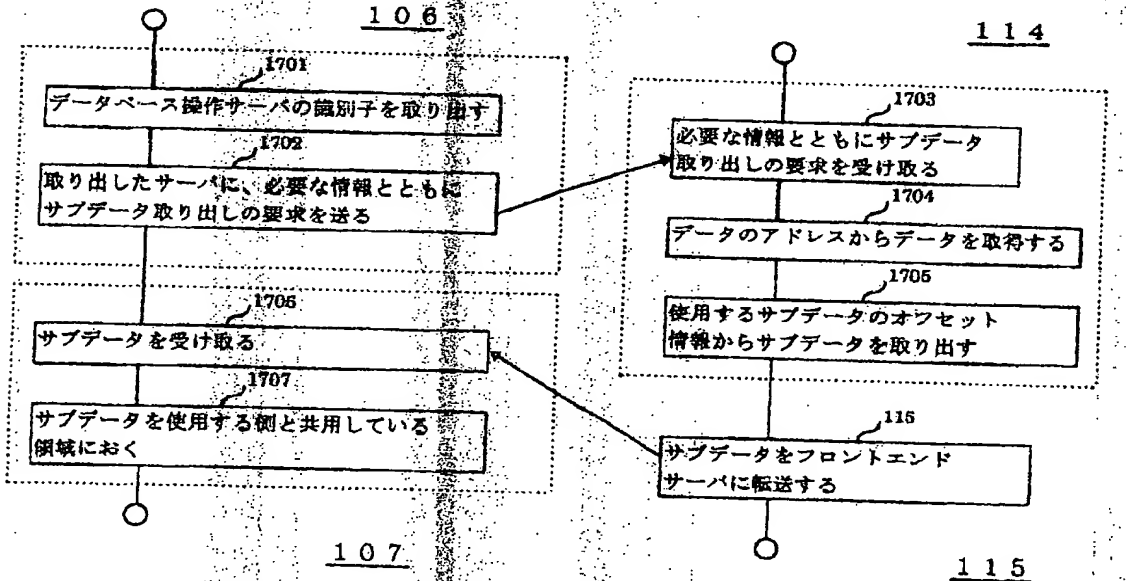
【図１６】

図１６


951
フロントエンドサーバ側
103
952
データベース操作サーバ側
113
1603
キューから結果を１つ取り出す
953
終了報告か？
yes
no
105
結果をUAPに送す
1604
実行した全ての
サーバから終了
報告はきたか？
no
yes
1602
結果はあるか？
yes
954
結果転送
終了報告転送
1601

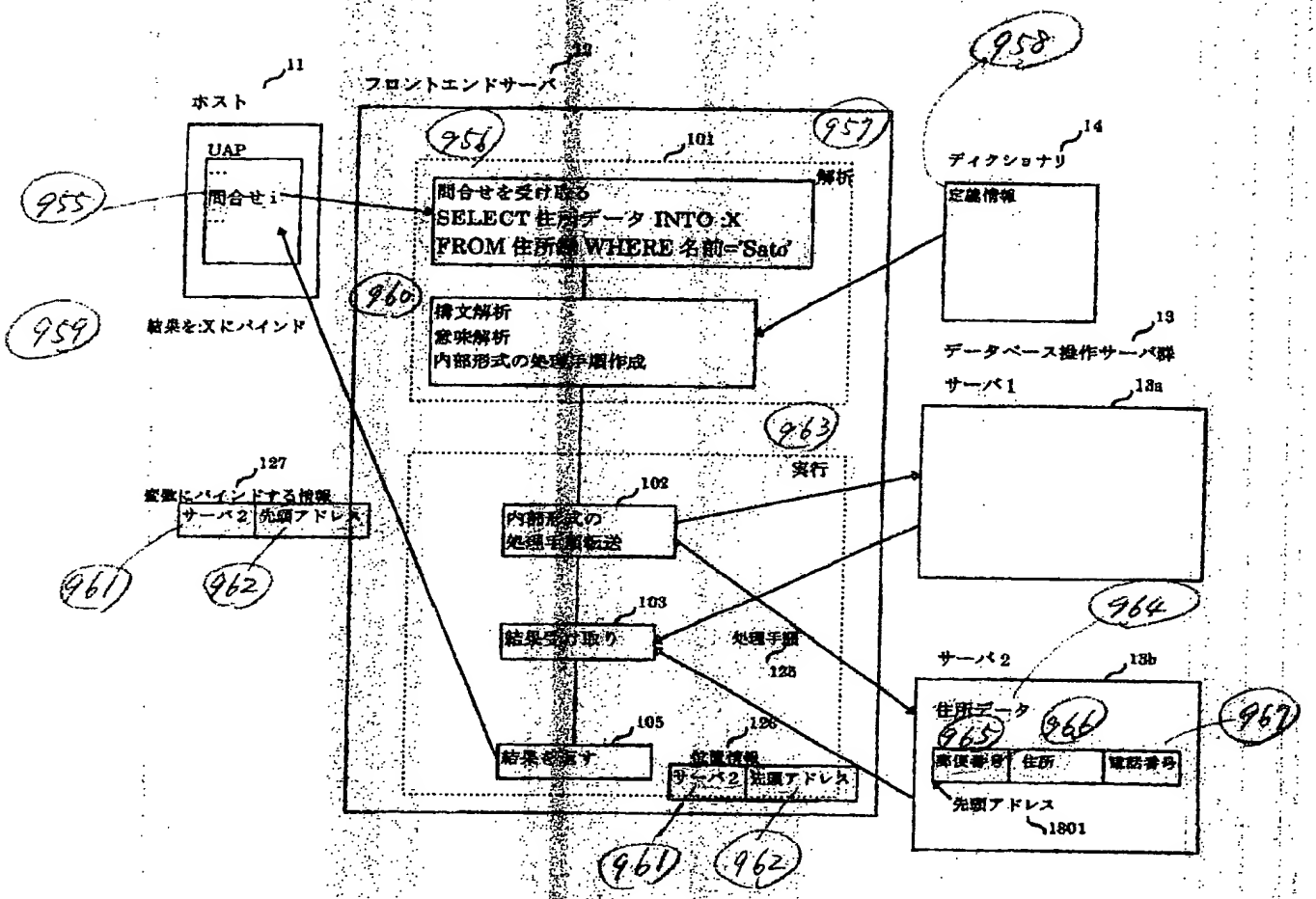
【図１７】

図１７


106
1701
データベース操作サーバの識別子を取り出す
1702
取り出したサーバに、必要な情報とともに
サブデータ取り出しの要求を送る
1705
サブデータを受け取る
1707
サブデータを使用する側と共用している
領域におく
107
114
1703
必要な情報とともにサブデータ
取り出しの要求を受け取る
1704
データのアドレスからデータを取得する
1705
使用するサブデータのオフセット
情報からサブデータを取り出す
115
サブデータをフロントエンド
サーバに転送する
115

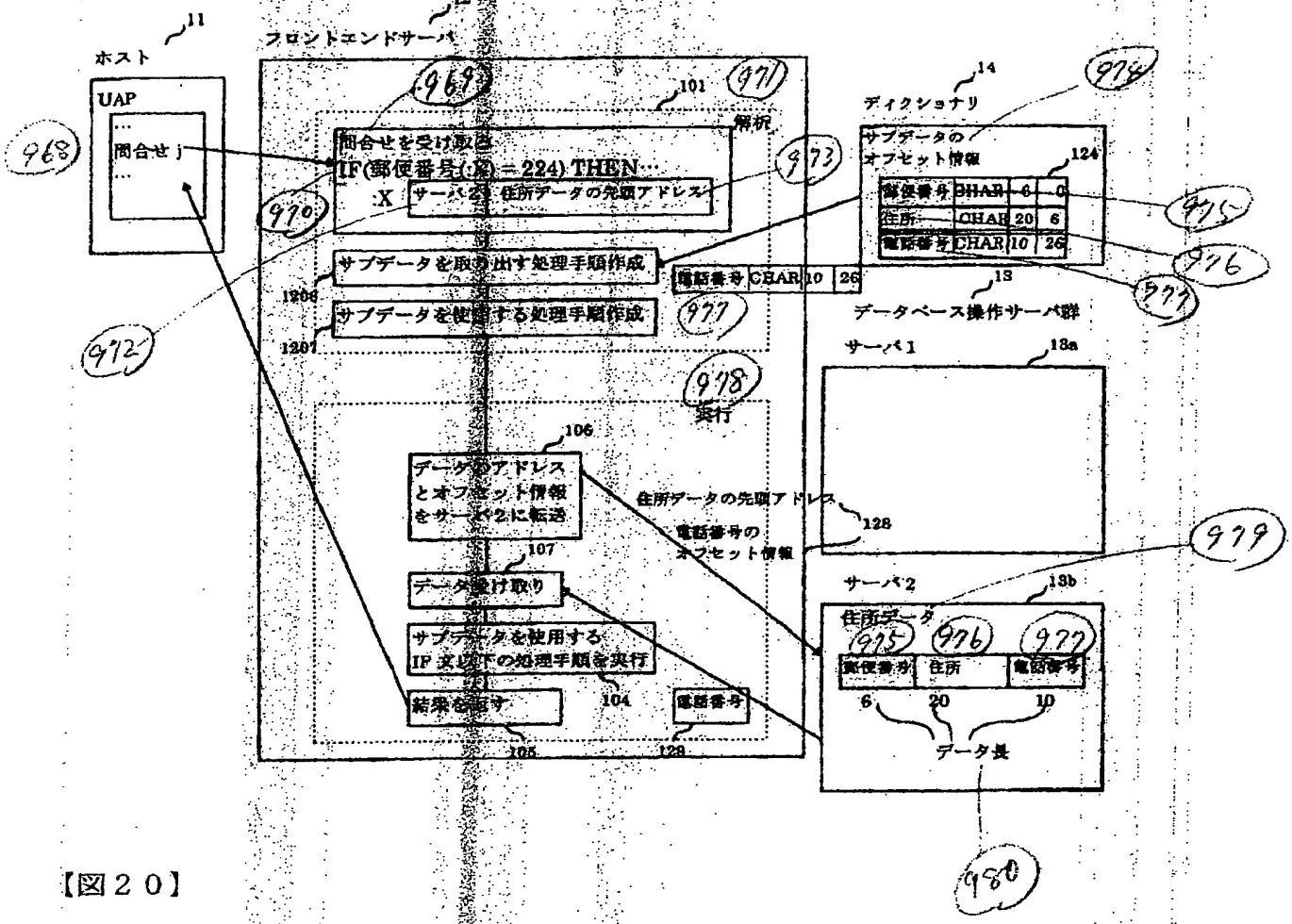
【図１８】

図１８


ホスト
11
フロントエンドサーバ
12
UAP
問合せ i
955
結果を:Xにバインド
959
956
957
101
問合せを受け取る
SELECT 住所データ INTO :X
FROM 住所録 WHERE 名前='Sato'
解析
960
構文解析
意味解析
内部形式の処理手順作成
963
実行
102
内部形式の
処理手順転送
103
結果受け取り
処理手順
125
105
結果を返す
126
位置情報
サーバ2
先頭アドレス
961
962
127
変数にバインドする情報
サーバ2
先頭アドレス
961
962
958
ディクショナリ
14
定義情報
データベース操作サーバ群
13
サーバ1
13a
964
サーバ2
13b
住所データ
965
966
967
郵便番号
住所
電話番号
先頭アドレス
1801

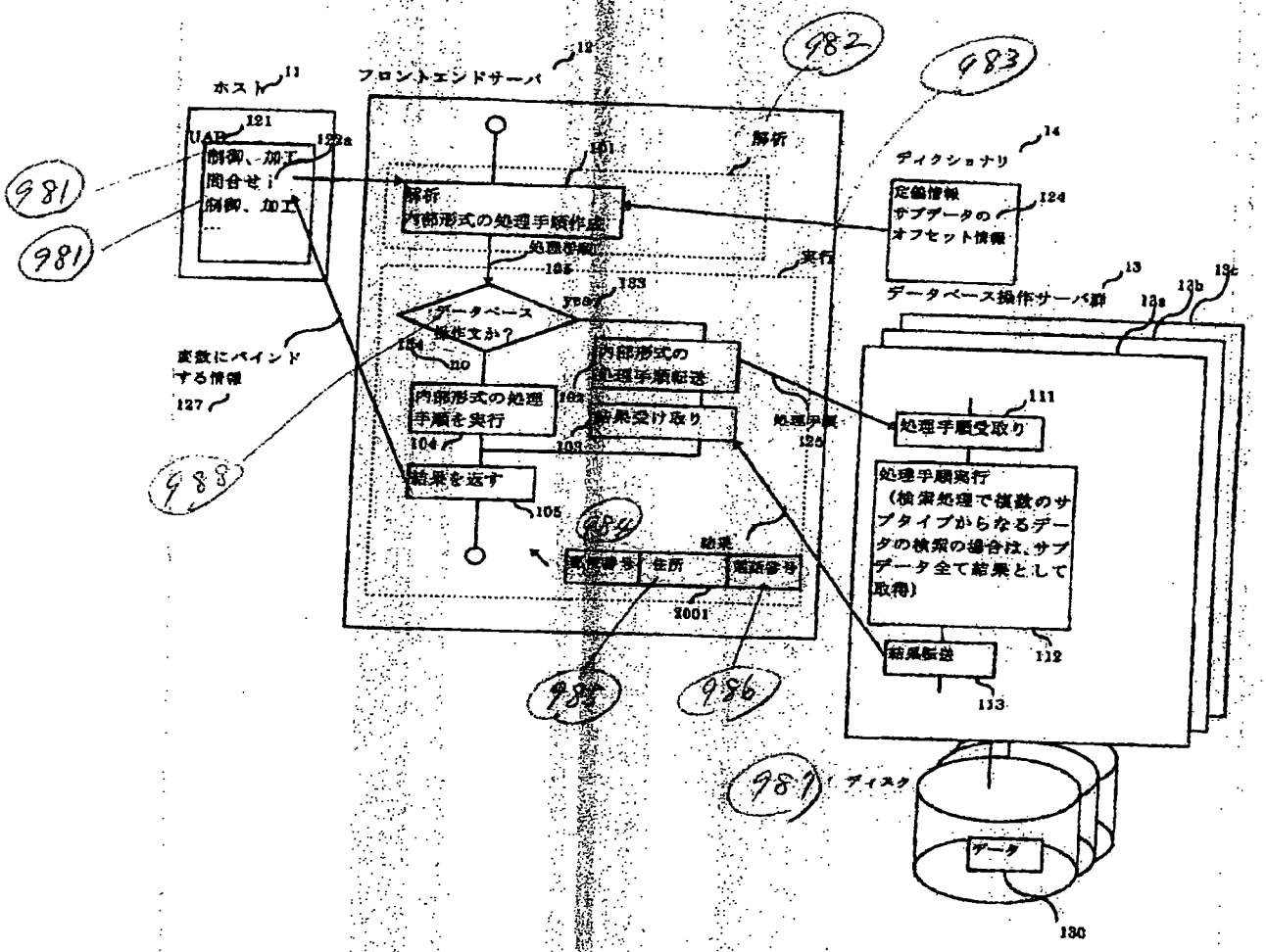
【図１９】

図１９


ホスト
11
UAP
問合せ j
フロントエンドサーバ
12
問合せを受け取る
IF(郵便番号(X) = 224) THEN…
X サーバ2 住所データの先頭アドレス
解析
101
サブデータを取り出す処理手順作成
1200
サブデータを使用する処理手順作成
1201
実行
データのアドレス
とオフセット情報
をサーバ2に転送
106
データ受け取り
107
サブデータを使用する
IF 文以下の処理手順を実行
結果を返す
104
105
電話番号
128
ディクショナリ
14
サブデータの
オフセット情報
124
郵便番号 CHAR 6 0
住所 CHAR 20 6
電話番号 CHAR 10 26
電話番号 CHAR 10 26
データベース操作サーバ群
13
サーバ1
13a
住所データの先頭アドレス
128
電話番号の
オフセット情報
サーバ2
13b
住所データ
郵便番号
住所
電話番号
6
20
10
データ長
968
969
970
971
972
973
974
975
976
977
978
979
980

【図２０】

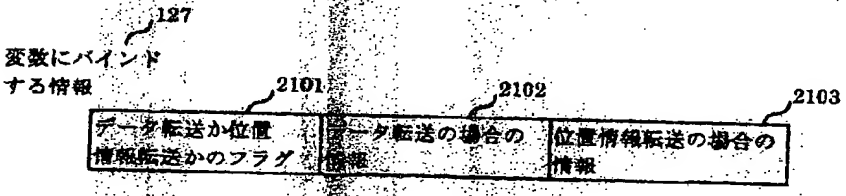
図２０



【図２１】

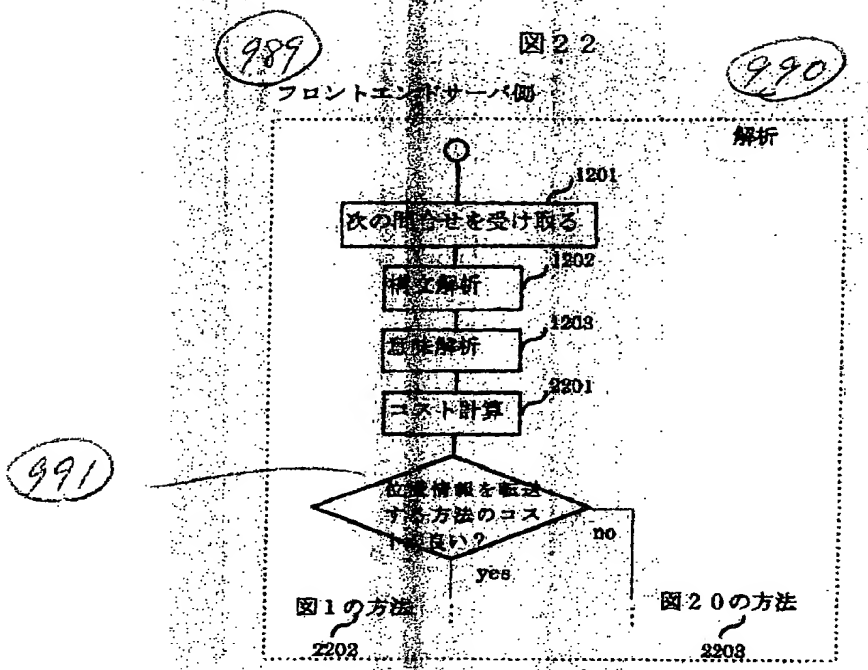
図２１

127



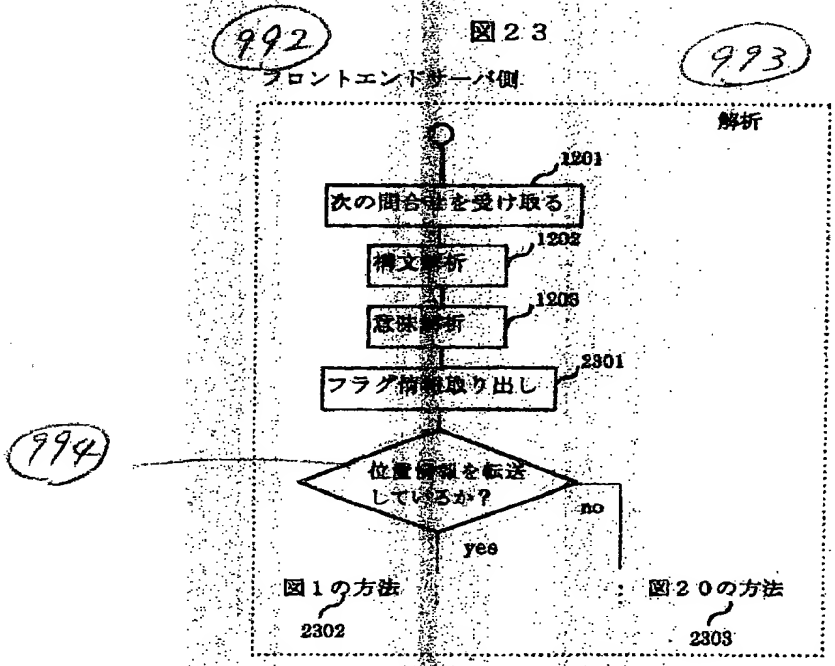
【図２２】

図２２


989
フロントエンドサーバ側
990
解析
1201
次の問合せを受け取る
1202
構文解析
1203
意味解析
2201
コスト計算
991
位置情報を転送する方法のコストが良い？
no
yes
図１の方法
2203
図２０の方法
2203

【図２３】

図２３


992
フロントエンドサーバ側
993
解析
1201
次の問合せを受け取る
1202
構文解析
1203
意味解析
2301
フラグ情報取り出し
994
位置情報を転送しているか？
no
yes
図１の方法
2302
図２０の方法
2303

This Page is inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☑ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLORED OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☑ REPERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning documents *will not* correct images problems checked, please do not report the problems to the IFW Image Problem Mailbox